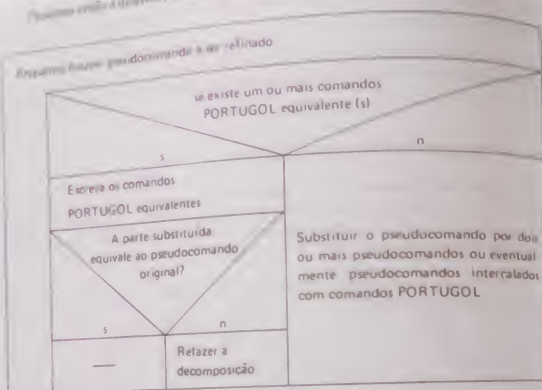


Podemos avaliar a qualidade da solução conforme o algoritmo abaixo



No nosso exemplo teríamos:

```

início | passo 2 |
      | ler dados |
      | processar  |
      | emitir relatório |
fim
    
```

Para o refinamento de um pseudocomando, utilizaremos a seguinte sintaxe:

```

refinamento de <pseudocomando>
  <pseudocomandos> ou <comandos PORTUGOL>
fim refinamento
    
```

O refinamento do pseudocomando "processar" será:

```

refinamento de "processar"
  CHAVE ← 1;
  enquanto CHAVE = 1 faça
    "efetuar alterações";
    "calcular vencimentos";
    "processar pedido";
    "emitir a mensagem";
  fim enquanto;
fim refinamento
    
```

o avança sucessivamente

```

refinamento de "efetuar alterações"
  CÓDIGO ← "A";
  então "modificar";
  então se CÓDIGO = "E"
    então "excluir";
    então se CÓDIGO = "I"
      então "incluir";
      então imprima ("ERRO");
    fim se;
  fim se;
fim se;
imprima (CÓDIGO);
fim refinamento
    
```

A metodologia de refinamentos sucessivos parte do princípio de que resolver um problema complexo é mais fácil se não precisamos considerar todos os aspectos do problema simultaneamente. A decomposição de um problema grande numa série de subproblemas mais simples até chegar a um nível onde pode-se tentar uma solução é o objetivo básico da técnica de refinamento sucessivo.

Quando um pseudocomando ou conjunto de pseudocomandos executam uma única função que concorre para a solução do problema, ele pode ser considerado um módulo funcional e poderá ser tratado como um procedimento ou uma função em PORTUGOL.

A técnica de modularização, associada aos refinamentos sucessivos, vai permitir a redução da complexidade dos problemas.

Um bom algoritmo deve procurar reduzir a interação entre módulos (acoplamento) e aumentar o relacionamento dos elementos de um mesmo módulo (coesão).

